Contrasting Different Definitions of DIV and MOD.

Written for the Comal80 Development Group
by Arne Christensen, Metanic ApS, March 1983

In a paper prepared for an earlier standardisation group, I surveyed the definitions of DIV and MOD made by two different Comal vendors, the original Pascal definition, and the mathematical definition of modulus. Of these four definitions only two distinct definitions could be distinguished. In the following I will describe these two definitions together with a third one and show where they differ.

First a bit of notation: I will distinguish the DIV and MOD operators of the two definitions by suffixing either letter a, letter b or letter c. So DIVa is different from DIVb.

Now, the three definitions are for integers (reals will be commented on separately):

X DIVa Y = the integer N which makes X - N*Y as small as possible while non-negative.

X MODa Y = X - (X DIVa Y)*Y

and

X DIVb Y = TRUNC (X/Y) ---truncation, which means throwing decimals away.

X MODb Y = X - (X DIVb Y)*Y

and

X DIVc Y = INT(X/Y) ---floor function

X MODc Y = X - (X DIVc Y)*Y

Notice, the similarities between the definitions of MODa, MODb and MODc.

I will show the difference by examples:

X	Y	X DIVa Y	X MODa Y	X DIVb Y	X MODb Y	X DIVc Y	X MODc Y
36	5	7	1	7	1	7	1
36	-5	-7	1	-7	1	-8	-4
-36	5	-8	4	-7	-1	-8	4
-36	-5	8	4	7	-1	7	-1
35	5	7	0	7	0	7	0
35	-5	-7	0	-7	0	-7	0
-35	5	-7	0	-7	0	-7	0
-35	-5	7	0	7	0	7	0

Here we note several interesting points which are also easily derived mathematically:

1. If X is a multiple of Y then DIVa, DIVb and DIVc give the same result and therefore also MODa, MODb and MODc. So, in this case the three definitions yield the same result.
2. This is also the case if X>0 and Y>0.
3. DIVb observes the normal sign conventions for division.
4. Neither DIVa nor DIVc observe the normal sign conventions for division. Of the two I find DIVa the most confusing.
5. All three MOD operators yield a result whose absolute values are in the range of 0 to ABS(Y)-1. The signs are as follows:
   MODa - the sign is always non-negative
   MODb - the sign follows the sign of X
   MODc - the sign follows the sign of Y

In our decision we must consider three points:

- What do we want the sign rules for DIV to be ?
  (If we want the normal sign rules then we must choose DIVb)
- What do we want the sign rules for MOD to be ?
- Do we want to keep the equation X = (X DIV Y)*Y + X MOD Y ?
  (If we do then we must choose EITHER DIVa and MODa OR DIVb and MODb OR DIVc and MODc.)

In the ISO definition of Pascal DIVb and MODa have been chosen. The MOD operator has been restricted to positive Y. When Y is positive MODa and MODc yield the same result, so it could also be said that MODc has been chosen. Note that as a consequence the equation X = (X DIV Y)*Y + X MOD Y does not always hold for negative X.

This choice differs from the original Pascal definition where DIVb and MODb were chosen.

It is interesting to note that in Ada DIVb and BOTH MODb and MODc (called REM and MOD in Ada) have been chosen.

With regard to real numbers we have basically two choises: We can use the definition without modification on the real numbers or we can round them to integers first, and then apply the definition.

The latter choice can prove useful because it will prevent small numerical errors from affecting the result.

I suppose that the final decision on this subject will have to await the decision on rounding vs. truncation of array indices.